Three Year Review

Dan Siewiorek
Carnegie Mellon University
Zary Segall, University of Oregon
December 1995

Contracts: "Fault Tolerant MACH"
N00014-92-J-4139
"Ultra-Dependable Real-Time Computing"
N00014-1-94-1-0210

1.0 Technical Objectives

1.1 Fault Tolerant MACH

The current trend in computing is towards open systems employing hardware and software from multiple vendors tied together by portable software packages. The UNIX operating system ushered in a new era of user freedom from proprietary hardware/software platforms that commercial vendors used to capture customers. UNIX provided a portable environment wherein a piece of software developed on one system could be moved to another system with minor effort. Since UNIX was available on a wide variety of platforms, the user could purchase the most cost effective hardware without incurring an enormous software redesign effort. MACH extends the UNIX portability beyond the hardware platform by providing a uniform treatment of both networked (often called distributed computing) and parallel processing (often called shared memory) computational models. MACH sets a trend for contemporary operating systems by employing a microkernel whereby the basic operating system functions, such as allocate memory or start up a task, are implemented in the microkernel. Traditional operating system services, such as a file system, are implemented as servers executing on top of the microkernel.

As computing systems assume more and more critical tasks wherein an error can have catastrophic consequences, attributes of computer systems other than just cost or performance become more important. One such attribute is the ability to tolerate a variety of errors ranging from physical defects to environmentally induced changes to human errors. There have been fault tolerant commercial computers for almost two decades. Most fault tolerant systems have involved proprietary hardware and software, locking users into a single vendor. Furthermore the user had to select either fault tolerance or performance - an application could not decide to place some resources on improving fault tolerance and the remaining resources on performance. No trade-off between fault tolerance and performance was possible. While a network of distributed computers running open system software has a natural degree of redundancy so that physical hardware failures could be tolerated, software to take advantage of this feature has been slow to develop. Research has produced software which can tolerate network node failure assume fail-fast network nodes, implying that faults are either detected or recovered from before erroneous output can enter the network. Current open systems such as MACH do not implement the fail-fast model.

DEPARTMENT OF THE NAVY
OFFICE OF NAVAL RESEARCH
SEATTLE REGIONAL OFFICE
1107 NE 45TH STREET, SUITE 350
SEATTLE WA 98105-4631

IN REPLY REFER TO:

4330
ONR 247
11 Jul 97

From: Director, Office of Naval Research, Seattle Regional Office, 1107 NE 45th St., Suite 350, Seattle, WA 98105
To: Defense Technical Center, Attn: P. Mawby, 8725 John J. Kingman Rd., Suite 0944, Ft. Belvoir, VA 22060-6218

Subj: RETURNED GRANTEE/CONTRACTOR TECHNICAL REPORTS

1. This confirms our conversations of 27 Feb 97 and 11 Jul 97. Enclosed are a number of technical reports which were returned to our agency for lack of clear distribution availability statement. This confirms that all reports are unclassified and are "APPROVED FOR PUBLIC RELEASE" with no restrictions.

2. Please contact me if you require additional information. My e-mail is *silverr@onr.navy.mil* and my phone is (206) 625-3196.

ROBERT J. SILVERMAN

CARNEGIE-MELLON UNIVERSITY

INTERNAL REPORT OF INVENTIONS AND PATENTS

Contract / Grant No. : **N00014-94-1-0210**
CMU Center Number: **1-51353**

Period of Contract **01/01/94** through **12/31/95**

I hereby certify that , to the best of my knowledge and belief, no inventions, improvements or discoveries, which reasonably appear to be patentable or actual patents were conceived or first actually reduced to practice by persons engaged in the performance of the work under the above contract / grant during the work period indicated, except as follows:

Name of Invention	Title of Invention

By Daniel P. Siewiorek
Title Professor
Date Feb. 29, 1996

The goal of this research was to design and implement a Fault Tolerant version of the MACH operating system (FT MACH) that adhered to the fail-fast model and allows the user to select the amount of fault tolerance (including none) to be allocated to each application.

1.2 Ultra-Dependable Real-Time Computing

Over the past 20 years, benchmarks have evolved from simple, synthetic programs to comprehensive application suites for measuring the performance of computer systems, both for users of systems and for designers of systems. The benchmarks have fostered a sense of competition among manufacturers to produce faster systems. Today there are no benchmarks to measure the robustness and dependability of computer systems. Without benchmarks it is difficult to compare the robustness and dependability of individual techniques or of complete systems. In addition, relative progress cannot be measured. The objective of Robustness Benchmarks is to define measures of robustness, develop methodologies for measuring robustness, and to implement portable software that can be used to evaluate fault tolerant systems.

2.0 Technical Approach

2.1 Fault Tolerant MACH

The initial focus was on adding error-detection mechanisms to various features of MACH. The first step added observability and controllability to services provided by the MACH run-time library. Library calls are made to an application built upon the microkernel. The library server has been modified so that all calls are encapsulated into a standard "envelop" providing a "flight record" of time, calling parameters, and returns. The envelop concept has been formalized as the sentry model. In this model, MACH services are viewed as a combination of all possible execution paths and data structures involved in serving a request. Sentries are placed at the entry and exit points of services in order to perform fault management. Hence, in the sentry model a MACH call can be guarded by more than one pair of entry/exit sentries. Sentries have been categorized to reflect their structure and functionality. Four types of sentries have been defined: Fault Detection Sentries (FDS), Fault Recovery Sentries (FRS), Fault Monitoring Sentries (FMS), and Validation/Fault Injection Sentries (VFS). Fault Monitoring Sentries have been implemented for user level operating system calls in MACH 3.0. These monitoring sentries report call entry and exit time stamps as well as input/output parameters. The ability to trace system behavior, particularly in the vicinity of an error, has been very useful at identifying software "bugs" that appeared under stressing workloads.

2.2 Ultra-Dependable Real-Time Computing

A methodology has been developed for the construction of user mode modular robustness benchmarks. The system is stressed with incorrect system calls representative of the type of errors made by application designers or corrupted data. The modular benchmarks focus on single errors to enhance repeatability and to isolate the corrupting input. The benchmarks are executed on the actual target system is contrast to fault injection which typically requires modifications to the system, simulation which is an imperfect model of the system, and physical methods such as heavy ion bombardment and pin-level injection which exposes systems to random errors and possible damage.

The Robustness Benchmarks target specific functions of the operating system (such as the memory allocator, the file system, the communication subsystem, the runtime library, etc.) and define a class of feasible faults (such as passing random characters that may have been

generated through communication line noise from remote computing sites) that are deemed most likely to occur with respect to that operating system feature. Each benchmark generates a series of test cases and keeps track of the number of cases which are successfully detected. The benchmark is robust enough to maintain accurate statistical count even if one of the tests crashes the operating system.

3.0 Accomplishments

3.1 Fault Tolerant MACH

The first Fault Recovery Sentry for MACH 3.0 implemented journalling. The Fault Monitoring Sentries are used to capture keyboard/mouse inputs as well as operating system call input/output parameters from application programs and to journal these parameters onto permanent stable storage. For typical interactive workstation user sessions (as opposed to compute-intensive workstation usage) journalling requires approximately 10 MBytes per hour of storage with CPU overheads ranging from a few percent to unnoticeable for applications such as word processing, drawing packages, and desktop publishing. Recovery after a crash is totally automatic and all data, except perhaps for the last keystroke, is automatically recovered through the replay of the journal. Journal replay time is a function of the amount of user interaction and the amount of computationally-intensive time. For interactive user-oriented sessions the replay time is typically around ten percent of the original session. Journalling Fault Recovery Sentries has been demonstrated with a wide variety of Unix-based applications and do not require detailed knowledge of the application's internal structure.

The second Fault Recovery Sentry for MACH 3.0 implemented check-pointing and rollback. A novel solution to capturing a checkpoint of multiple concurrent tasks coupled with journalling reduce the amount stable storage requirements to a total of 10 MBytes and recovery time to a few minutes with check-pointing occurring as a background activity.

3.2 Ultra-Dependable Real-Time Computing

Based upon our previous study of Robustness Benchmarks the technology was applied, under separate funding, to an Air Force Satellite computer ASCM based upon the 1750A instruction set by the IBM Federal Systems Division at Manassas, Virginia. Using a systematic, modular approach the parameters for operating system calls were identified as well as likely error manifestations. A watchdog program executed a series of operating system calls with a variety of the illegal parameters. Almost 20,000 tests were executed resulting in dozens of cases causing warm restarts of computer modules and one case of a cold restart. Approximately one-fourth of the operating system calls were thus tested in this ADA environment. The Fault Monitoring Sentries have been used to observe MACH 3.0 system behavior prior to a benchmark induced fault. If the fault results in a system crash, we attempt to generalize the system state that induced the crash so that a robustness benchmark can be designed to probe that single feature.

The Robustness Benchmark methodology has been applied to an aerospace fault tolerant computer. It has work has been extended and ported to test the Mach operating system. Data is currently being collected and a paper will be written shortly. A kernel-level fault injection and test environment is being implemented utilizing the Sentry mechanism. This environment will be compared to the Mach Robustness Benchmark results.

4.0 Importance of the Accomplishments

The concept of Sentries has been defined, designed, implemented, and demonstrated. Sentries are implemented as middleware between unmodified application code and unmodified operating systems. Sentries intercept operating system service requests from the application. Sentries can provide services both on entry to the operating system and upon exiting back to the application. Services provided by the sentries enhance the observability and controllability of the system. Several classes of services have been identified including: journalling for roll-back, assertions for error detection, replication for fault tolerance, fault injection for validation, etc.

Sentries represent a framework for producing highly-dependable systems from commercial off-the-shelf hardware and software. Unmodified legacy application software can be turned into fault-tolerant services. The sentry mechanism has been demonstrated with journalling applied to the Mach operating system for workstations and the Windows operating system for personal computers. Journalling sentries allow complete recovery of even multitasking legacy application software from errors induced by hardware (e.g. power outage), software (undoing a system call which led to a system crash), and operator mistakes (e.g. undoing the previous command).

The Robustness Benchmark methodology has been effective at discovering design flaws in error detection/handling mechanisms is both commercial and dedicated aerospace fault-tolerant systems.

5.0 Transitions of Research

5.1 To Navy and DOD Organizations

5.2 To Industry

The commercial relevance of the Sentry technology, supported under these ONR grants, has been recognized through the award by ARPA of a SBIR to Systems Technology/Development Corporation (ST/DC) of Reston, Virginia. Initial results during Phase I of the SBIR have demonstrated that Sentry technology provides highly effective levels of fault tolerance, with one to two orders of magnitude reduction in costs compared to proprietary hardware/software solutions. Unlike most commercial fault tolerant systems, which require users to modify and recompile their application code, the application transparency features of Sentries achieves fault tolerance capabilities without modifying application code. A commercial product based on the sentry technology is planned for the first quarter of 1996. Negotiations are underway with two developers/distributors of PC based software for the Sentry based commercial product. Symantec, Dynamics, and Martin Marietta have expressed firm interest to help transition this technology into their current products and applications.

6.0 Research Papers

Gupta, A. P., W. P. Birmingham, D. P. Siewiorek, "Automating the Design of Computer Systems," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 12, No. 4, pp, 473-487, April 1993.

Siewiorek, Daniel P., Asim Smailagic, "A Case Study in Embedded-System Design" The VuMan 2 Wearable Computer," IEEE Design and Test of Computers, Vol. 10, No. 4, 1993.

Russinovich, M. Z. Segall, D. P. Siewiorek, "Application Transparent Fault Management in Fault-Tolerant MACH", Proceedings 23 International Symposium on Fault Tolerant Computing, Toulouse, France, pp. 10-19, June 1993.

Siewiorek, D. P., J.J. Hudak, B.-H. Suh, Z. Segall, "Development of a Benchmark to Measure System Robustness", Proceedings 23 International Symposium on Fault Tolerant Computing, Toulouse, France, pp. 88-97, June 1993.

Hudak, J., B. Suh, D. Siewiorek, Z. Segall, "Evaluation and Comparison of Fault-Tolerant Software Techniques", IEEE Transactions on Reliability, Vol. 42, No. 2, pp.. 190-204, June 1993.

Mukherjee, A., "Measuring Software Dependability by Robustness Benchmarking", Technical Report CMU-CS-94-148, May 1994.

Russinovich, M., "Application-transparent fault management", PhD Dissertation, Electrical and Computer Engineering, Carnegie Mellon University, August 1994.

Russinovich, M. and Z. Segall, "Application-Transparent Check-pointing in Mach 3.0/UX", 27th Hawaii Int. Con. System. Sciences, Jan. 1995.

Dingman, C. P., J. Marshall, D. P. Siewiorek, "Measuring Robustness of a Fault Tolerant Aerospace System," Proceedings 25 International Symposium on Fault Tolerant Computing, Los Angles, CA, pp. 522-527, June 1995

Russinovich, M., Z. Segall, "Fault-Tolerance for Off-The-Shelf Applications and Hardware", Proceedings 25 International Symposium on Fault Tolerant Computing, Los Angles, CA. pp 67-71, June 1995.

Siewiorek, D., "Niche Successes to Ubiquitous Invisibility: Fault-Tolerant Computing Past, Present, and Future", Special Silver Jubilee Proceedings, International Symposium on Fault Tolerant Computing, Los Angles, CA., June 1995.

Christopher Dingman, Joseph Marshall, Daniel Siewiorek, "Measuring Robustness from the System Call Level", 4th IEEE International Workshop on Evaluation Techniques for Dependable Systems, San Antonio, TX, October 1995